Trigger Neden Kullanılır?

Trigger, veritabanında belirli bir olayın gerçekleştiğinde otomatik olarak tetiklenen bir veritabanı nesnesidir. Trigger'lar, veritabanı yönetim sistemlerinde kullanıcı tarafından belirlenen bir olay veya koşul gerçekleştiğinde otomatik olarak çalışacak işlemleri gerçekleştirmek için kullanılır.

Trigger'ların kullanılmasının bazı yaygın nedenleri şunlardır:

* Veri Bütünlüğü: Trigger'lar, veri bütünlüğünü korumak için kullanılabilir. Örneğin, bir tabloya yeni bir kayıt eklendiğinde veya güncellendiğinde, ilgili başka bir tablodaki verileri otomatik olarak güncelleyebilir veya kontrol edebilir.
* Veri Doğrulama: Trigger'lar, veritabanına eklenen veya güncellenen verilerin belirli kurallara veya koşullara uygun olduğunu doğrulamak için kullanılabilir. Örneğin, bir trigger, bir çalışanın maaşının belirli bir aralıkta olması gerektiğini kontrol edebilir veya bir siparişin geçerli bir müşteriye ait olduğunu doğrulayabilir.
* Otomatik Loglama: Trigger'lar, belirli bir olay gerçekleştiğinde otomatik olarak loglama işlemlerini gerçekleştirebilir. Örneğin, bir tabloya yeni bir kayıt eklediğinizde, bir trigger otomatik olarak bu olayı bir log tablosuna kaydedebilir.
* İş Akışı Yönetimi: Trigger'lar, veritabanında belirli bir iş akışını yönetmek için kullanılabilir. Örneğin, bir sipariş tablosuna yeni bir sipariş eklendiğinde, bir trigger otomatik olarak siparişin onay sürecini başlatabilir veya ilgili kullanıcılara bildirim gönderebilir.
* Veri Denetimi: Trigger'lar, veritabanında belirli verilerin değiştirilmesini veya silinmesini önlemek için kullanılabilir. Örneğin, bir tablodaki önemli verilerin yanlışlıkla silinmesini önlemek için bir trigger kullanılabilir.

Trigger Olumlu Yönleri

* Veri Bütünlüğünü Sağlama: Trigger'lar, veri bütünlüğünü korumak için kullanılabilir. Örneğin, bir tabloya yeni bir kayıt eklenirken veya güncellenirken, trigger otomatik olarak belirli koşullara veya kural setlerine uyan verileri kontrol edebilir. Bu, veri tutarlılığını sağlayarak hatalı veya uygunsuz veri girişlerini engeller.
* Veri Doğrulama ve Kontrol: Trigger'lar, veri doğrulama ve kontrol kurallarını otomatik olarak uygulayabilir. Örneğin, bir trigger, bir işlem gerçekleştiğinde verilerin belirli koşullara uygun olup olmadığını kontrol edebilir. Bu, veri kalitesini artırır ve hatalı veri girişlerini engeller.
* Otomatik İşlemler: Trigger'lar, belirli bir olay gerçekleştiğinde otomatik olarak işlemleri tetikleyebilir. Örneğin, bir tabloya yeni bir kayıt eklendiğinde, trigger otomatik olarak ilgili tablolarda veya verilerde değişiklik yapabilir veya başka bir işlemi tetikleyebilir. Bu, veritabanında daha karmaşık iş mantığı veya iş akışı senaryolarını uygulamak için kullanılabilir.
* Veri Denetimi: Trigger'lar, belirli veri değişikliklerini denetlemek için kullanılabilir. Örneğin, bir tablodan veri silme veya güncelleme işlemi yaparken trigger, belirli kurallara uymayan veya kısıtlamalara aykırı olan işlemleri engelleyebilir. Bu, veri bütünlüğünü korur ve veri güvenliğini sağlar.
* Otomatik Loglama ve İzleme: Trigger'lar, belirli olayları otomatik olarak loglayabilir veya izleyebilir. Örneğin, bir tabloya yeni bir kayıt eklediğinizde, trigger otomatik olarak bu olayı bir log tablosuna kaydedebilir. Bu, izlenebilirlik ve denetlenebilirlik açısından faydalıdır.

Trigger Olumsuz Yönleri

* Beklenmeyen Etki: Trigger'lar, bir olay gerçekleştiğinde otomatik olarak çalışan kod parçaları olduğu için dikkatlice tasarlanmalıdır. Yanlış veya hatalı bir trigger, beklenmeyen etkilere neden olabilir. Örneğin, yanlış bir trigger, veritabanı performansını olumsuz etkileyebilir veya beklenmeyen veri değişiklikleri yapabilir.
* Performans Sorunları: Trigger'lar, bir veritabanı işlemi gerçekleştiğinde otomatik olarak tetiklendiğinden, fazla sayıda veya karmaşık trigger'lar veritabanı performansını olumsuz etkileyebilir. Trigger'lar, veri tabanında yazma veya okuma işlemleri gerçekleştirirken performans kaybına neden olabilir.
* Karmaşıklık ve Bakım Zorlukları: Trigger'lar, veritabanı mantığını karmaşıklaştırabilir ve bakımı zorlaştırabilir. Birden fazla trigger'ın etkileşimi veya trigger zincirleri oluşturulması durumunda, kodun anlaşılması ve sürdürülmesi zorlaşabilir. Bu, gelecekteki değişiklikler veya güncellemeler için zaman ve çaba gerektirebilir.
* Yanlış Kullanım veya Kötü Tasarım: Trigger'lar, doğru kullanılmazsa veya kötü bir şekilde tasarlanırsa, beklenmedik sonuçlara yol açabilir. Yanlış bir tetikleyici, yanlış veya gereksiz veri değişikliklerine neden olabilir veya uygunsuz bir işlemi tetikleyebilir. Ayrıca, yanlış kullanılan trigger'lar, veri bütünlüğünü veya doğruluğunu etkileyebilir.
* İzleme ve Hata Ayıklama Zorlukları: Trigger'lar, otomatik olarak çalıştıkları için izlenmesi ve hata ayıklanması zor olabilir. Trigger'lar, karmaşık bir iş mantığı veya iş akışını içerebilir ve hata ayıklama süreci karmaşık olabilir.

Trigger Örneği

CREATE TRIGGER update\_total\_spent

AFTER INSERT ON orders

FOR EACH ROW

BEGIN

UPDATE customers

SET total\_spent = total\_spent + NEW.order\_amount

WHERE customer\_id = NEW.customer\_id;

END;

Yukarıdaki trigger, "orders" tablosuna yeni bir satır eklendiğinde (**AFTER INSERT ON orders**), her bir satır için (**FOR EACH ROW**) "customers" tablosundaki ilgili müşterinin "total\_spent" değerini güncelleyecektir. **NEW** kavramı, yeni eklenen siparişi temsil eden bir değişkeni ifade eder.

Python ile veri tabanı bağlantısı yapıldıktan sonra trigger kullanıldığı durumda python içerisinde bunu nasıl çağırabiliriz/nasıl bir mantık kurulabilir?

Python'da trigger'ları çalıştırmak yerine, veritabanına sorgular göndererek trigger'ları tetikleyebilirsiniz. Bu şekilde, veritabanına yapılan değişiklikler trigger'ları otomatik olarak tetikleyecektir.

import sqlite3

# Veritabanı bağlantısını oluştur

conn = sqlite3.connect('mydatabase.db')

cursor = conn.cursor()

# Veritabanına sorgular göndererek trigger'ı tetikle

cursor.execute('''

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_amount)

VALUES (1, 1, '2023-07-07', 100.00)

''')

# Veritabanı değişikliklerini kaydet

conn.commit()

# Veritabanı bağlantısını kapat

conn.close()

Yukarıdaki Python kodu, **orders** tablosuna yeni bir kayıt ekleyerek trigger'ı tetiklemektedir. Bu sorgu, **mydatabase.db** adlı SQLite veritabanında yer alan **orders** tablosuna yeni bir kayıt ekleyecektir. Trigger, bu ekleme işlemiyle otomatik olarak tetiklenecek ve ilgili müşterinin **customers** tablosundaki **total\_spent** değeri güncellenecektir.

Python kodunda veritabanı bağlantısı oluşturulduktan sonra, ilgili tabloya veya trigger'ı tetiklemek için veritabanına sorgular gönderebilirsiniz. Sorgular veritabanında değişiklik yapacak veya trigger'ı tetikleyecek işlemleri gerçekleştirmek için kullanılır.